Server(Prover/Signer)-Aided Verification of Identity Proofs and Signatures

Chae Hoon Lim and Pil Joong Lee

Department of Electrical Engineering
Pohang University of Science and Technology (POSTECH)
Pohang, 790-784, KOREA
E-mail : lch@baekdu.postech.ac.kr ; pjl@cipher.postech.ac.kr

Abstract. Discrete log based identification and signature schemes are well-suited to identity proof and signature generation, but not suitable for verification, by smart cards, due to their highly asymmetric computational load between the prover/signer and the verifier. In this paper, we present very efficient and practical protocols for fast verification in these schemes, where the verifier with limited computing power performs its computation fast with the aid of the powerful prover/signer. The proposed protocols require very small amounts of computation and communication. The prover/signer only needs to perform a few modular exponentiations in real-time and the two interacting parties only need to communicate a few long numbers. Using the proposed prover-aided verification (PAV) protocol, the verifier can perform the Schnorr-like identification scheme almost as fast as the Guillou-Quisquater scheme. We generalize the PAV protocol into the signer-aided verification (SAV) protocol, which can be used for verification of any public function.

1 Introduction

Based on zero-knowledge proof techniques, a lot of identification and digital signature schemes have been developed [1-6]. Among them, Schnorr-like schemes [4-6] are particularly attractive for use in smart cards or other environments with limited computing power, since the prover/signer needs almost no8 fax. real-time computation with preprocessing/precomputation techniques [4,7-9]. However, verification requires exponentiation involving a lot of multiplications, which is disadvantageous compared to Fiat-Shamir-like schemes [1-3]. This asymmetric computational load may restrict applications of these schemes, when implemented on a weak power device such as a smart card, into environments where only one-way proofs are sufficient. Thus what is further desired for these schemes

L.C. Guillou and J.-J. Quisquater (Eds.): Advances in Cryptology - EUROCRYPT '95, LNCS 921, pp. 64-78, 1995.

would be that proofs in the other way are also efficient for smart card implementations. This motivated us to develop methods for speeding up the computation by the verifier in some way or another.

We first considered the applicability of the server-aided approach to secret computation, first proposed by Matsumoto et al. [10] and since then widely studied by many researchers [11-15], to the public verification of identity proofs and signatures. And we found a related work performed by Yen and Laih [16], but unfortunately their protocol can be easily shown to be insecure. Furthermore, this kind of protocols seems to need too much amount of communication to be practical for smart card applications. There are fundamentally different requirements for the server-aided secret computation protocol and the server-aided public verification protocol. The former has to guarantee the security of the involved secret information, while the latter requires the assurance of the integrity of computation results returned from the server. This difference of requirements seems make the latter protocol needlessly complicated and hard to guarantee the correctness of the required verification.

In this paper we propose efficient protocols for speeding up the verification of identity proofs and signatures. The key idea is to use the precomputation based on a fixed base element and then mirror the action of the prover/signer. In the proposed protocols, we assume that the proving/signing terminal is much more powerful than the verifying device so that it can perform several exponentiations in real-time. This situation will commonly arise in a smart card based system when a powerful terminal proves to a smart card or when signature verification is performed on a smart card. Thus the resulting protocols may be called as prover/signer-aided verification (PAV/SAV) protocols since the verifier performs the required verification fast by borrowing the computing power of the prover/signer.

Compared to the protocol for server-aided RSA computation, our proposed protocols are much more efficient and practical since only a few long numbers need to be exchanged and only a few modular exponentiations need to be performed by the prover/signer. For example, using the proposed prover-aided verification (PAV) protocol, the verifier can execute the Schnorr-like identification scheme [4-6] almost as fast as the Guillou-Quisquater scheme [2], only with exchange of one long number and without loss of security. This will make Schnorr-like identification schemes much more attractive for smart card implementations since now smart cards can also perform the required verification fast. By generalizing the PAV protocol, we also present a signer-aided verification (SAV) protocol with which the verifier can check the validity of signatures with any desired convincing probability. For example, if a convincing probability of $1 - 2^{-t}$ is acceptable in a real-time protocol, a signature generated by Schnorr's scheme can be verified in about $3t$ multiplications on average. Finally we show that the proposed techniques can be used for verification of any public function by presenting a fully generalized version of the server-aided verification protocol.

2 Prover-Aided Verification of Identity

Throughout this paper, we will use the following conventions, unless otherwise stated. Let p and q be two large public primes such that q divides $p-1$ and g be an element of order q in Z_p. We denote the bit-length of p (q, resp.) by n (l, resp.) (i.e., $|p| = n, |q| = l$). Let (s, v) be the secret and public key pair of the prover/signer, where $v = g^{-s} \bmod p$ with $s \in Z_q$. We assume that precomputation of random powers to the fixed base g is performed in advance and thus does not take time during the protocol execution.

The computation of $a^x b^y \bmod p$ with $|x| = l$ and $|y| = t$ is assumed to be carried out by the square-and-multiply algorithm with a precomputed value of $ab \bmod p$ (see [4]). We assume that the most significant bits of the exponents, x and y, are always one for completeness, though their effect on the performance is negligible. Then the above computation can be completed in $1.5l + 0.25(t-1)$ multiplications for $l > t$, and $1.75l - 0.75$ for $l = t$, on average. Multiplication will always denote multiplication mod p and multiplication mod q will be neglected when counting the number of multiplications.

2.1 PAV Protocol for Schnorr's Identification Scheme

The following is the five-move protocol for prover-aided verification in Schnorr's identification scheme [4]. Here t is a parameter that determines the security level of the identification scheme, usually lying between 20 and 40.

0) (Preprocessing) The prover picks a random number $r \in Z_q$ and computes $x = g^r \bmod p$. Similarly the verifier computes $z = g^{-K} \bmod p$ with randomly chosen K over Z_q.
1) The prover sends x to the verifier.
2) The verifier randomly picks an integer $e \in [0, 2^t)$ and sends it to the prover.
3) The prover computes and sends $y = r + se \bmod q$.
4) The verifier randomly picks an integer $k \in [0, 2^t)$, computes and sends $u = (K + y)k^{-1} \bmod q$.
5) The prover computes $w = g^u \bmod p$ and sends it back to the verifier.
6) Finally the verifier checks if the following equation holds :

$$x = w^k v^e z \bmod p \tag{1}$$

Note that for security the precomputed value z should not be revealed to the prover at least until the protocol is completed. This must be observed in every protocol presented in this paper. If desired, the computation of $k^{-1} \bmod q$ in step 4) may be performed in the preprocessing stage. Steps 1) - 3) exactly correspond to the original Schnorr scheme whose verification equation equals

$$x = g^y v^e \bmod p. \tag{2}$$

On the other hand, steps 4) - 6) correspond to the protocol in which the verifier computes $g^y \bmod p$ with the aid of the prover. Thus we can see that by borrowing the prover's computing power the verifier can reduce the computational load of l bit exponentiation to that of t bit exponentiation.

Security : Equation (1) shows that the values of x, v and e, which are determined in the first half of the protocol, cannot be modified, without knowledge of k, in the latter half. On the other hand, the value of u, which is the only data available to the prover for extracting information on the secret number k, releases no information on k since even z is not available at this point. As a result, the prover may guess k but has no way to verify its guess. The above two facts show that the PAV protocol is unconditionally sound since no information on k is released in the Shannon-theoretic sense and since without knowing k the dishonest prover cannot convince the verifier with more than guessing probability.

There may be a slight advantage on the prover's side. Throughout the whole protocol, the prover is given two chances of cheating the verifier : either by guessing e in step 1) as in the original Schnorr scheme or by guessing k in step 5). The latter guess can be successful independently of the former guess since, once the former is turned out to be wrong from the response of step 2), the prover knows how to manipulate w to pass the verification of step 6), of course, under the assumption that its guess at k is correct. Thus the added steps 4) and 5) only gives the prover another chance of random guessing. This will be of little value to the (dishonest) prover. Consequently, we conclude that the prover-aided approach to fast verification preserves almost the same security level of the original scheme.

Efficiency : The verifier can check the verification equation (1) in about $1.75t + 0.25$ multiplications on average. This is almost the same amount of computation as is required in the GQ scheme. Note that with the original verification equation (2), about $1.5l + 0.25(t-1)$ multiplications are required. For example, with $l = 160$ and $t = 20$, the equality of equation (1) can be checked in 35.25 multiplications, while validating equation (2) requires 244.75 multiplications, on average. Thus about 210 multiplications can be saved in this case using the proposed verification protocol.

The above efficiency is obtained only by increasing the number of communication bits by $n+l$. The computational complexity imposed on the prover is also very small, just one exponentiation ($1.5(l-1)$ multiplications on average). No restriction on the computing power of the prover will be necessary due to this increase of computational amount, since such computation can be carried out in real-time even on the PC (personal computer). Therefore, in typical smart card-based systems, we will be able to obtain great computational advantage using the PAV protocol only with a small increase of communication.

2.2 PAV Protocol for Brickell-McCurley's Scheme

Brickell and McCurley [5] modified the Schnorr scheme in order to enhance the security at the cost of more computation and communication. The basic differences are that all exponents are selected and computed modulo $p-1$ rather than modulo q and that q is kept secret from the users (so the modulus p should be chosen such that $p-1$ is hard to factor). The resulting protocol can be proven to be secure, assuming that $p-1$ is hard to factor, and remains as secure as the Schnorr scheme even if $p-1$ is factored.

The PAV protocol for the Brickell-McCurley (BM) scheme is the same as that for the Schnorr scheme, except that all arithmetics on exponents should be done modulo $p-1$. Thus the performance improvement by the PAV protocol is much more drastic in this scheme. For example, with $n = 512$ and $t = 20$, the original verification requires 772.75 multiplications on average, while the prover-aided verification still requires 35.25 multiplications. This amounts to more than a twenty four-fold improvement. Since main disadvantage of the BM scheme can be eliminated with the PAV protocol, the BM scheme may be preferred to the Schnorr scheme in view of security.

We finally would like to mention that the PAV protocol does not affect the provable security of the original scheme since no additional information on the secret key of the prover is involved in the prover-aided verification part. Note that a three-move identification scheme is said to be secure (in the sense of Feige-Fiat-Shamir [17]) if the protocol execution releases no useful information on the prover's secret.

2.3 PAV Protocol for Okamoto's Scheme

Okamoto [6] has proposed another modification of Schnorr's scheme with the feature of provable security. Since it is somewhat different from the Schnorr scheme in basic construction, we describe his scheme together with the proposed verification protocol. Let p and q be as before and g_1 and g_2 be elements of order q in Z_p. The public key of the prover in the Okamoto scheme is $v = g_1^{-s_1} g_2^{-s_2} \bmod p$, where s_1 and s_2 in Z_q are his secret keys. The PAV protocol for Okamoto's scheme is as follows.

0) (Preprocessing) The prover randomly picks $r_1, r_2 \in Z_q$ and computes $x = g_1^{r_1} g_2^{r_2} \bmod p$. Similarly the verifier computes $z = g_1^{-K_1} g_2^{-K_2} \bmod p$ with $K_1, K_2 \in Z_q$.
1) The prover sends x to the verifier.
2) The verifier randomly picks an integer $e \in [0, 2^t)$ and sends it to the prover.
3) The prover computes $y_1 = r_1 + s_1 e \bmod q$ and $y_2 = r_2 + s_2 e \bmod q$ and sends them to the verifier.
4) The verifier randomly picks $k \in [0, 2^t)$, computes $u_1 = (K_1 + y_1)k^{-1} \bmod q$ and $u_2 = (K_2 + y_2)k^{-1} \bmod q$, and sends them back to the prover.

5) The prover computes $w = g_1^{u_1} g_2^{u_2} \bmod p$ and sends it to the verifier.
6) Finally the verifier checks if the following equation holds :

$$x = w^k v^e z \bmod p \tag{3}$$

Though the Okamoto scheme is somewhat different from the Schnorr scheme, we can see that the performance of the PAV protocol remains almost the same. Compare the above equation (3) with the original verification equation :

$$x = g_1^{y_1} g_2^{y_2} v^e \bmod p \tag{4}$$

The only difference is that in the above the verifier computes $g_1^{y_1} g_2^{y_2} \bmod p$ as $w^k z \bmod p$ with the aid of the prover. Note that it is unnecessary to use different values of k to compute u_1 and u_2 due to the involvement of distinct random secrets, K_1 and K_2, of l bit size (In any case, knowing one small random secret will be sufficient to cheat the verifier).

Table 1 below summarizes the performance of three identification schemes and their PAV versions. The certificate for the public key v is not taken into account when counting the number of communication bits and the computational amounts for preprocessing are also excluded. The number of multiplications is counted for the average case. Finally, note that we are using the parameters n, l and t as $n = |p|, l = |q|$ and $t = |e| = |k|$, respectively.

			Schnorr	Brickell et al.	Okamoto
Orig	Mul	P	almost 0	almost 0	almost 0
		V	$1.5l + 0.25(t-1)$	$1.5n + 0.25(t-1)$	$1.75l + 0.125t + 2.37$
	Comm		$2n + l + t$	$3n + t$	$2n + 2l + t$
PAV	Mul	P	$1.5(l-1)$	$1.5(n-1)$	$1.75l + 0.25$
		V	$1.75t + 0.25$	$1.75t + 0.25$	$1.75t + 0.25$
	Comm		$3n + 2l + t$	$5n + t$	$3n + 4l + t$

Table 1. Performance of PAV protocols for three identification schemes

Finally we note that the proposed PAV protocol can also be adapted for identification schemes with composite moduli. For example, in Girault's modification of the Schnorr scheme based on composite discrete logarithms [18], the order of the based element g is made public and thus the PAV protocol for Schnorr's scheme can be applied directly. On the other hand, in the similar protocol using the self-certified public key [19], the based element g has a maximal order modulo a composite and the signature component y is not reduced modulo any number. Thus it is not feasible to compute multiplicative inverses of exponents. For this scheme, the verifier may first raise both sides of the verification equation to the k-th power and then apply the PAV protocol (or it may use the protocol to be presented in section 3). Of course, the performance will be somewhat degraded in this case.

3 Signer-Aided Verification of Signatures

There exists the same asymmetry of computational load in digital signature schemes derived from identification schemes based on the discrete logarithm problem. Thus these signatures are easy to generate but hard to verify with smart cards. This section is devoted to developing an efficient protocol for signer-aided verification of signatures. Of course, the role of the powerful server need not be assumed by the signer itself in this case. Since typical application of this protocol will be signature verification on the smart card, the server may be a powerful terminal with which the smart card interacts.

We only explain the proposed SAV protocol with Schnorr's signature scheme, but it can be used for verification of other signature schemes based on the discrete logarithm problem as well (e.g., see [20-23] for generalized ElGamal-type signature schemes and their message recovery variants). In fact, the proposed technique can be applied to server-aided verification of any public function, as will be illustrated in the next section.

3.1 SAV Protocol for Schnorr's Signature Scheme

For the moment, let us suppose that the signer's public key $v = g^{-s} \bmod p$ is globally known and frequently used (this may be the case if we have to frequently verify signatures of some central authorities). Then we can adapt the PAV protocol into the SAV (signer-aided verification) protocol as follows, where h denotes a one-way hash function producing randomly and uniformly c-bit digests (see below).

0) (Preprocessing) The verifier computes $z = g^{-K_1} v^{-K_2} \bmod p$ with $K_1, K_2 \in Z_q$.
1) The signer sends the signature $\{x, y, m\}$ to the verifier, where $x = g^r \bmod p$ and $y = r + se \bmod q$ with $e = h(x, m)$.
2) The verifier computes $e = h(x, m)$. Then it randomly picks an integer $k \in (0, 2^t]$, computes $u_1 = (K_1 + y)k^{-1} \bmod q$ and $u_2 = (K_2 + e)k^{-1} \bmod q$, and sends them to the signer.
3) The signer computes and sends $w = g^{u_1} v^{u_2} \bmod p$.
4) The verifier then checks if $x = w^k z \bmod p$ holds. If the check succeeds, the verifier accepts and stores $\{e, y\}$ as a valid signature for message m.

We first want to note that the length of hash-values used in any signature schemes should be at least 128 bits, contrary to the minimal length of 64 or 72 bits that many researchers (e.g., see [1,2,4]) suggested. This is because the signer can find two different messages with the same signature using the birthday paradox if short hash-values are used. If such a thing is feasible, then the signer may deny later the signature of one message by presenting the other message with the same signature. This situation is essentially the same, as far as the

legality of signature is concerned, as the case where an outside attacker finds two different messages with the same hash-value, obtains a signature for the message favorable to the signer and then claims that the signer signed the other message favorable to himself.

A slight modification may achieve the same effect that can be obtained by the use of longer hash-values without increasing the computational load of the verifier, but this does not matter in the current SAV protocol. From now on, we will assume that hash-values are randomly distributed over Z_q (i.e., $c = l = |q|$) as in the DSS [24].

The above SAV protocol achieves a security level of 2^{-t}. The signer cannot use in step 3) a value of v different from the one publicly known or sent in step 1), due to its involvement in the computation of z. Other security considerations are the same as in the PAV protocol. Thus, a fake signature can be made to be accepted only when the guess of k is correct. If a false acceptance with probability of 10^{-6} can be tolerated in a real-time protocol, then the signature can be verified in 29.5 multiplications on average. However, this protocol seems not practical in general, since the precomputation using the signer's public key is not possible in most cases. Thus the above SAV protocol needs to be augmented by somewhat different technique.

The problem we are faced with is to compute the part $v^e \bmod p$ of the verification equation $x = g^y v^e \bmod p$ with the aid of the signer, where the signer's public key v is assumed to vary in every run of the protocol. Our solution is to blind the public key v by raising to the k-th power and then multiplying by a random power of g, i.e., form $u = g^K v^k \bmod p$ ($K \in Z_q, k \in (0, 2^t]$), so that the signer, no matter how powerful it is, cannot deduce k from u (and thus cannot modify v) with more than the guessing probability of 2^{-t}. For this, the verifier must compute $v^k \bmod p$ before beginning the signer-aided verification, which increases the verifier's computational load almost twice compared to the above case. The following is the final SAV protocol for Schnorr's signature scheme.

0) (Preprocessing) The verifier computes $z_1 = g^{-K_1} \bmod p$ and $z_2 = g^{-K_2} \bmod p$ with $K_1, K_2 \in Z_q$.
1) The signer sends the signature $\{x, y, m\}$ to the verifier, where $x = g^r \bmod p$ and $y = r + se \bmod q$ with $e = h(x, m)$.
2) The verifier randomly picks an integer $k \in (0, 2^t]$ and computes $u_1 = z_1 v^k \bmod p$ using the signer's public key v. The verifier also computes $u_2 = (K_2 + ky + K_1 e) \bmod q$ with $e = h(x, m)$ and sends u_1 and u_2 to the signer.
3) The signer computes and sends $w = u_1^e g^{u_2} \bmod p$.
4) Finally the verifier checks that $x^k = w z_2 \bmod p$. If the equation holds, the verifier accepts and stores $\{e, y\}$ as a valid signature for message m.

The above verification is based on the following identity :

$$x^k = (g^{-K_1} v^k)^e \cdot g^{K_2 + K_1 e + ky} \cdot g^{-K_2} \bmod p \qquad (5)$$

Note that since the value of e computed as $e = h(x, m)$ by the verifier is embedded in u_2, it is of no use for the signer to use a different value of e when computing w in step 3). The on-line computational load for the verifier is about $3t-1$ multiplications on average. Thus, with a convincing probability of $1-10^{-6}$, the verifier can validate a signature in 59 multiplications on average. This is a substantial improvement over direct verification requiring about 279.25 multiplications, if a small probability of false acceptance can be tolerated. If more strict verification is required, we may choose $t = 30$, in which case the signature can be verified in 89 multiplications with probability of false acceptance of 10^{-9}.

It is interesting to note that the SAV protocol may be viewed as an interactive proof system for language membership [25], though the proof is trivial, where the language L consists of a set of valid signatures generated with the Schnorr scheme, i.e.

$$L = \{(x, y, m, v) | x = g^y v^e \bmod p \text{ with } e = h(x, m)\}. \tag{6}$$

In the SAV protocol, the verifier with limited computing power wants to be convinced that a given instance belongs to L. The above discussion shows that the SAV protocol satisfies the two conditions of an IP system, completeness and soundness.

The following table shows the performance of the proposed SAV protocol for Schnorr's signature scheme. Here we assume that the hash-value e is of l bit size. The message m and the public key certificate are not included in the number of communication bits.

		Original	SAV
Mul	Signer	almost 0	$1.75l - 0.75$
	Verifier	$1.75l - 0.75$	$3t - 1$
Commun		$n + 2l$	$4n + 2l$

Table 2. Performance of SAV protocol for Schnorr's signature scheme

3.2 Batch SAV Protocol for Schnorr's Scheme

A collection of signatures can be verified more efficiently by processing in a batch. Naccache et al. [26] presented (interactive and probabilistic) batch verification protocols for DSA at Eurocrypt'94, together with several other useful techniques to improve the performance of DSA (but the interactive batch verification protocol was shown to be insecure [27]).

Let $\{x_i, y_i\}$, for $i = 1, 2, \cdots, N$, be Schnorr's signatures for messages m_i signed by the same signer, where $x_i = g^{r_i} \bmod p$ and $y_i = r_i + se_i \bmod q$

with $e_i = h(x_i, m_i)$. Then the verifier can check the validity of the signatures by batch-processing with the equation

$$\prod_{i=1}^{N} x_i^{k_i} = g^{\sum_{i=1}^{N} k_i y_i} \cdot v^{\sum_{i=1}^{N} k_i e_i} \bmod p, \tag{7}$$

where k_i's are random numbers of t-bit size chosen by the verifier. The parameter t determines the level of confidence for batch verification.

We first explain a method for efficiently evaluating the left-hand side of equation (6) using the idea from [8]. It can be computed by arranging the N terms of small powers into a groups consisting of b terms, preparing all products of possible combinations among b terms in each group and then applying the square-and-multiply algorithm. We can then show that the required computation can be completed in $\frac{2^b-1}{2^b}(t-1)a + t + (2^b - b)a - 2$ multiplications on average. For this, we also need a storage for $(2^b - 1)a$ values.

Table 3 below summarizes, for some selected parameters, the numbers of multiplications and storage required for the computation of the left-hand side of equation (6) using this method. From the table, we can see that if the verifying device is equipped with sufficient storage, a number of signatures can be verified with great efficiency. Batch verification on the PC may be such a case. For example, 16 signatures generated by the same signer can be validated in about 464 multiplications on average, where $t = 30$ is assumed and 279.25 multiplications for computing the right-hand side of equation (6) are included.

N		2	3	4	6		8		12		16
(a, b)		(1,2)	(1,3)	(2,2)	(3,2)	(2,3)	(4,2)	(2,4)	(4,3)	(3,4)	(4,4)
Storage		3	7	6	9	14	12	30	28	45	60
Mul	$t = 20$	34.3	39.6	50.5	66.8	61.3	83.0	77.6	104.5	107.4	137.3
	$t = 30$	51.8	58.4	75.5	99.3	88.8	123.0	106.4	149.5	145.6	184.8

Table 3. Resource requirements for computing the left-hand side of equ. (6)

Now, let us consider the batch verification on the smart card. Since typical smart cards under current technology do not have much storage, a relatively small number of signatures can be processed at a time. In this case, the computation of the right-hand side of equation (6) seems a quite heavy load to the smart card. Thus we may use a batch SAV protocol for this computation. Let us consider the following protocol.

0) (Preprocessing) The verifier computes $z_1 = g^{-K_1} \bmod p$ and $z_2 = g^{-K_2} \bmod p$ with $K_1, K_2 \in Z_q$.
1) The signer sends $\{x_i, y_i, m_i\}$ $(1 \leq i \leq N)$ to the verifier.

2) The verifier first computes $e_i = h(x_i, m_i)$. Then it randomly picks $N+1$ integers k_i, for $i = 0, 1, \cdots, N$, over $(0, 2^t]$, and then computes $u_1 = z_1 v^{k_0} \bmod p$ and u_2, u_3 as

$$u_2 = k_0^{-1} \sum_{i=1}^{N} k_i e_i \bmod q, \quad u_3 = K_2 + K_1 u_2 + \sum_{i=1}^{N} k_i y_i \bmod q.$$

The verifier then sends u_1, u_2 and u_3 to the signer.

3) The signer computes and sends $w = u_1^{u_2} g^{u_3} \bmod p$.

4) Finally the verifier checks if the following equation holds :

$$\prod_{i=1}^{N} x_i^{k_i} = w z_2 \bmod p \tag{8}$$

If it holds, the verifier accepts and stores $\{e_i, y_i\}$ as valid signatures for messages m_i for $i = 1, 2, \cdots, N$.

The above batch verification is based on the following identity :

$$\prod_{i=1}^{N} x_i^{k_i} = (g^{-K_1} v^{k_0})^{k_0^{-1} \sum_{i=1}^{N} k_i e_i} \cdot g^{k_0^{-1} K_1 \sum_{i=1}^{N} k_i e_i + \sum_{i=1}^{N} k_i y_i + K_2} \bmod p \tag{9}$$

Using the above batch SAV protocol, the verifier can compute the right-hand side of equation (6) in $1.5t - 0.5$ multiplications on average if we neglect the arithmetics mod q. Therefore, we can verify, for example, four signatures in about 80 multiplications on the smart card, with a convincing probability of $1 - 10^{-6}$ ($t = 20$), if the smart card has a scratch pad memory for ten values or so. Note that if different signers are involved, each signer's public key must be blinded individually and thus the performance will be degraded. But this is also the case for direct verification.

The batch SAV protocol has one undesirable property, compared to the SAV protocol of the previous subsection, in the sense that its security is dependent upon the computing power of the signer. That is, for small N, the signer may try to find the random secret numbers k_i's from the value of u_2 by an exhaustive search using the birthday paradox. This is clearly undesirable but seems inevitable due to the involvement of secret numbers in the exponent of v.

From the equation $u_2 k_0 + \sum_{i=1}^{N/2} k_i e_i = \sum_{i=1+N/2}^{N} k_i e_i \bmod q$ where we assume that N is even, k_i's can be computed in $L \log_2 L$ operations with $L = 2^{t(1+N/2)}$. For example, for $N = 2$ and $t = 20$, we have $L = 2^{40}$. However, such an attack can be mounted only after u_2 is given. Thus it is unlikely that this attack makes any practical threat to the protocol even for the above minimal parameters, since it is infeasible to perform 2^{40} operations in a second or so. Other security considerations are the same as in the SAV protocol.

4 Server-Aided Verification of General Functions

We now present a fully generalized version of server-aided verification protocols which can be used for verification of any public function. Suppose that the verifier, with the aid of a powerful server, wants to check the equality of the following general equation defined over a finite group G :

$$y^{\beta} = \prod_{i=1}^{N} x_i^{\alpha_i} \tag{10}$$

All involved elements are assumed to be public and variable. The following protocol allows the verifier to test the equality of the above equation with a convincing probability of $1 - 2^{-t}$.

0) (Preprocessing) The verifier randomly picks an element $g \in G$ and computes $z_i = g^{K_i}$ with $K_i, \in G$ for $i = 0, 1, \cdots, N$.
1) The verifier randomly picks an integer $k \in (0, 2^t]$ and then computes the following values :

$$u_0 = z_0 y^k, \quad u_i = z_i x_i^k \ (1 \leq i \leq N), \quad u_{N+1} = K_0\beta - \sum_{i=1}^{N} K_i\alpha_i + K_{N+1}$$

Then the verifier sends $\{g, u_i, \alpha_i, \beta\}$ to the server.
2) The server computes and sends the following value :

$$w = g^{u_{N+1}} u_0^{-\beta} \prod_{i=1}^{N} u_i^{\alpha_i}$$

3) Finally the verifier checks if $z_{N+1} = w$ holds.

The above server-aided verification is based on the following identity :

$$g^{K_{N+1}} = g^{K_0\beta - \sum_{i=1}^{N} K_i\alpha_i + K_{N+1}} \cdot (g^{K_0} y^k)^{-\beta} \cdot \prod_{i=1}^{N} (g^{K_i} x_i^k)^{\alpha_i} \tag{11}$$

The element g may be globally fixed and, if the group order $|G|$ is known, all the exponents can be reduced modulo $|G|$. The protocol achieves a security level of 2^{-t} since the only way to cheat the verifier is to guess k and manipulate y and/or x_i. The number of group multiplications required of the verifier is around $(1.5t - 0.5)(N + 1)$ on the average. If there are M fixed elements in equation (9), this quantity can be reduced to $(1.5t - 0.5)(N - M + 1)$.

All the protocols presented so far are special cases of the above protocol. Note that with t-bit randomizers (blinding factors), signature schemes involving a fixed base element can be verified in $3t - 1$ multiplications while the other schemes such as Guillou-Quisquater [2] and Ohta-Okamoto [3] can be verified in about $4.5t - 1.5$ multiplications on average. Even for the GQ scheme, this is a

considerable improvement over direct verification in case where a moderate level of confidence is sufficient (e.g., 88.5 vs 223.25 for $t = 20$ and 128 bit hash-values).

The above server-aided approach to fast verification will be useful for most public key cryptographic schemes when executed between two parties with asymmetric computing power. Typical applications may be found in the interactive protocols between smart cards and terminals. Since the proposed protocol is independent of the size of exponents and its security level is independent of the server's power, the advance of cryptanalytic methods (based either on software or on hardware) will never adversely affect its performance. Rather, the performance may be further improved in case that the size of group order is increased.

5 Summary and Conclusion

We have presented an elegant way to speed up the computation by the verifier in discrete logarithm-based identification schemes (Schnorr, Brickell-McCurley, Okamoto, etc.), with the aid of the powerful prover. The proposed prover-aided verification (PAV) protocol is secure and efficient : Only with a small amount of additional communication and with almost the same level of security as the original scheme, the verifier can perform the Schnorr-like identification scheme almost as fast as the Guillou-Quisquater scheme. In particular, the efficiency of the proposed protocol is independent of the size of exponents and thus Brickell-McCurley's scheme may be preferred to the Schnorr scheme due to its enhanced security. The proposed PAV protocol will make Schnorr-like identification schemes much more attractive for smart card implementations since now smart cards can also perform the required verification fast.

By generalizing the PAV protocol, we have also presented a signer-aided verification (SAV) protocol that can be adapted for verification of any public function. The proposed SAV protocol is also quite efficient in both computation and communication. With a convincing probability of $1 - 2^{-t}$, the validity of a signature can be checked in about $3t$ multiplications on average for discrete logarithm-based schemes and in about $4.5t$ multiplications on average for the GQ scheme. The batch SAV protocol enables more efficient verification of a collection of signatures.

The proposed server-aided verification protocol will be useful for many public key cryptographic schemes carried out between users with asymmetric computing powers. Smart card verification of identity proofs and signatures will be one of the most attractive application areas of the protocol. Another important application can be found in designing efficient protocols for authenticated key exchange between smart cards and servers (computers) (see [28]).

Finally we would like to mention that if the communication cost is relatively low, we can considerably reduce the computational complexity for the SAV protocol by adapting the server-aided approach for RSA computation (e.g., see [29]). Of course, in this case, its security relies on the computing power of the server as in the batch SAV protocol presented in this paper.

References

1. A.Fiat and A.Shamir : 'How to prove yourself : Practical solution to identification and signature problems', *Advances in Cryptology-Crypto'86*, Springer-Verlag, pp.186-194 (1988).
2. L.C.Guillou and J.J.Quisquater : 'A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory', *Advances in Cryptology-Eurocrypt'88*, Springer-Verlag, pp.123-128 (1988).
3. K.Ohta and T.Okamoto : 'A modification of the Fiat-Shamir scheme', *Advances in Cryptology-Crypto'88*, Springer-Verlag, pp.232-243 (1990).
4. C.P.Schnorr : 'Efficient signature generation by smart cards', *Journal of Cryptology*, 4(3), pp.161-174 (1991).
5. E.F.Brickell and K.S.McCurley : 'An interactive identification scheme based on discrete logarithm and factoring', *Journal of Cryptology*, 5(1), pp.29-39 (1992).
6. T.Okamoto : 'Provably secure and practical identification schemes and corresponding signature schemes', *Advances in Cryptology-Crypto'92*, Springer-Verlag, pp.31-53 (1993).
7. E.F.Brickell, D.M.Gordon, K.S.McCurley and D.B.Wilson : 'Fast exponentiation with precomputation', *Advances in Cryptology-Eurocrypt'92*, Springer-Verlag, pp.200-207 (1993).
8. C.H.Lim and P.J.Lee : 'More flexible exponentiation with precomputation', *Advances in Cryptology-Crypto'94*, Springer-Verlag, pp.95-107 (1994).
9. P.de Rooij : 'Efficient exponentiation using precomputation and vector addition chains', In *Pre-proceedings of Eurocrypt'94*, pp.403-416 (1994).
10. T.Matsumoto, K.Kato and H.Imai : 'Speeding up secret computations with insecure auxiliary devices', *Advances in Cryptology-Crypto'88*, Springer-Verlag, pp.497-506 (1990).
11. J.J.Quisquater and M.De Soete : 'Speeding up smart card RSA computation with insecure coprocessors', In *Proc. Smart Card 2000*, North-Holland, 191-197 (1991).
12. B.Pfitzmann and M.Waidner : 'Attacks on protocols for server-aided RSA computation', *Advances in Cryptology-Eurocrypt'92*, Springer-Verlag, pp.153-162 (1993).
13. T.Matsumoto, H.Imai, C.S.Laih and S.M.Yen : 'On verifiable implicit asking protocols for RSA computation', *Advances in Cryptology-Auscrypt'92*, Springer-Verlag, pp.296-308 (1993).
14. S.Kawamura and A.Shimbo : 'Fast server-aided secret computation protocols for modular exponentiation', *IEEE J. Selected Areas in Commun.*, 11(5), 778-784 (1993).
15. J.Burns and C.J.Mitchell : 'Parameter selection for server-aided RSA computation schemes', *IEEE Trans. Computers*, 43(2), 163-174 (1994).
16. S.M.Yen and C.S.Laih : 'Server-aided honest computation for cryptographic applications', *Computers Math. Applic.*, 26(12), pp.61-64 (1993).
17. U.Feige, A.Fiat and A.Shamir : 'Zero-knowledge proofs of identity', *J. Cryptology*, 1(2), pp.77-94 (1988).
18. M.Girault : 'An identity-based identification scheme based on discrete logarithms modulo a composite number', *Advances in Cryptology-Eurocrypt'90*, Springer-Verlag, pp.481-486 (1991).

19. M.Girault : 'Self-certificated public keys', *Advances in Eurocrypt'91*, Springer-Verlag, pp.490-497 (1991).
20. K.Nyberg and R.Rueppel : 'Message recovery for signature schemes based on the discrete logarithm problem', submitted to *Designs, Codes and Cryptography* (also appears in *Pre-proceedings of Eurocrypt'94*).
21. P.Horster, H.Petersen and M.Michels : 'Meta-ElGamal signature schemes', In *Proceedings of 2nd ACM Conference on Computer and Communication Security* (1994).
22. P.Horster, H.Petersen and M.Michels : 'Meta message recovery and meta blinded signature schemes based on the discrete logarithm problem and their applications', In *Pre-Proceedings of Asiacrypt'94*, pp.185-196 (1994).
23. L.Harn and Y.Xu : 'Design of generalized ElGamal type digital signature schemes based on discrete logarithm', *Electronics Letters*, 30(24), pp.2025-2026 (1994).
24. NIST : 'Digital signature standard', *FIPS PUB 186* (1994).
25. S.Goldwasser, S.Micali and C.Rackoff : 'The knowledge complexity of interactive proof systems', *SIAM J. Comput.*, 18(1), pp.186-208 (1989).
26. D.Naccache, D.M'raihi, D.Raphaeli and S.Vaudenay : 'Can D.S.A. be improved ? -Complexity trade-offs with the digital signature standard', In *Pre-proceedings of Eurocrypt'94* (1994).
27. C.H.Lim and P.J.Lee : 'Security of interactive DSA batch verification', *Electronics Letters*, 30(19), pp.1592-1593 (1994).
28. C.H.Lim and P.J.Lee : 'Fast authenticated key exchange with the aid of the communicating partner', in preparation (available from the authors by e-mail).
29. C.H.Lim and P.J.Lee : 'Signer-aided probabilistic verification of digital signatures using random decomposition', in preparation (available from the authors by e-mail).